|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_Коновалов Илья Николаевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_\_\_\_ИУ5-15Б\_\_\_\_\_\_\_\_\_\_\_\_\_

Тип практики ЛАБОРАТОРНО-ВЫЧИСЛИТЕЛЬНЫЙ ПРАКТИКУМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия КАФЕДРА «СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ» (ИУ5) МГТУ им. Н.Э.Баумана

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Коновалов И.Н.**\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Папшев И.С.**\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2021 г.*

**СОДЕРЖАНИЕ**

1. Цели и задачи практики.
2. Индивидуальное задание.
3. Листинг демонстрационной программы
4. Заключение.
5. Список использованной литературы
6. Приложение (одно на бригаду)  
   Листинг разработанной библиотеки функций для контроля и обработки вводимого с клавиатуры текста с необходимыми для её использования комментариями.

***Введение***

**1. Цели и задачи практики**

Задачами учебной практики «Лабораторно-вычислительный практикум» являются:

* Разработка на языке С++ библиотеки функций для решения задач индивидуальных заданий Лабораторно-вычислительного практикума.
* Использование при выполнении заданий методов групповой разработки программ

***Основная часть***

Разработка программного продукта выполняется бригадой с использованием методов групповой разработки программ.

Состав бригады (3 студента):

1. Самойлов Константин – руководитель группы, разработчик, тестировщик
2. Бенц Ян – разработчик, тестировщик
3. Коновалов Илья – разработчик, тестировщик

***Техническое задание:***

Ввести с клавиатуры текст, состоящий из нескольких предложений. Текст состоит из латинских (или русских) букв (строчных и прописных), пробелов и знаков пунктуации (точки, запятые и др.) и заканчивается символом "конец текста" (смотри таблицу кодов ASCII). Предложения в тексте разделяются точками. Количество предложений и их содержание должно обеспечивать проверку правильности выполнения задания, то есть являться тестом для разработанной программы.

БЕЗ ИСПОЛЬЗОВАНИЯ типа STRING:

разработать функцию для ввода такого текста с клавиатуры. Функция должна контролировать ввод, чтобы вводить только указанные в задании символы и возвращать строку с введенным текстом;

разработать структуры Text и Sentence для хранения введенного текста, которые обеспечивали бы возможность обращения к символам текста как к элементам трехмерного массива, индексами которого являются номер предложения в тексте, номер слова в предложении и номер символа в слове;

разработать функцию, которая преобразует введенный текст в массив из предложений;

разработать функцию, которая преобразует предложение в массив из слов;

поместить определения этих функций и структур, которые являются базовыми для ДЗ (так как могут использоваться для выполнения любого варианта задания), в отдельный модуль (файл) и подключить его к программе;

заполнить структуру Text введенным текстом;

выполнить в соответствии с индивидуальным заданием преобразование данных в структуре Text;

распечатать преобразованный текст. Печать предложения начинать с новой строки, слова печатать через один пробел, сохраняя в предложении использованные знаки пунктуации.

**2. Индивидуальное задание.**

2.1. В составе группы мною была разработана архитектура проекта, прототипы и реализация некоторых функций, в частности «Индивидуального задания» варианта №14: оставить в предложениях слова, отличающиеся от последнего слова предложения, в которые каждая буква входит не менее двух раз.

2.2. Разработана демонстрационная программа.

Листинг программы приведен в п.3 отчета.

1. **Листинг демонстрационной программы**

Мое индивидуальное задание:

void variant15(Sentence& sentence) {  
// Оставить в предложениях слова,  
// отличающиеся от последнего слова предложения,  
// в которых гласные буквы (a, e, i, o, u, y) чередуются с согласными.  
  
 int LastWordIndex = sentence.size - 1;  
 wchar\_t\*\* newWords = new wchar\_t\*[sentence.size];  
 wchar\_t\* currentWord;  
 int newSize = 0;  
 bool isCorrect;  
 bool prevSoglas = true;  
 bool currSoglas = true;  
 char arr[] = {'a','e','i','o','u','y'};  
  
 for (int i = 0; i <= LastWordIndex; i++) {  
 if (wcscmp(sentence.words[LastWordIndex], sentence.words[i]) || i == LastWordIndex) {  
 isCorrect = true;  
 currentWord = sentence.words[i];  
 for (int f = 0; f < sizeof(arr)/sizeof(arr[0]); f++) {  
 if (currentWord[0] == arr[f]) {  
 prevSoglas = false;  
 }  
 }  
 for (int j = 1; j < wcslen(currentWord) - 1; j++) {  
 for (int k = 0; k < sizeof(arr)/sizeof(arr[0]); k++) {  
 if (currentWord[j] == arr[k]) {  
 currSoglas = false;  
 }  
 }  
 if (currSoglas == prevSoglas) {  
 isCorrect = false;  
 break;  
 }  
 else {  
 prevSoglas = currSoglas;  
 currSoglas = true;  
 }  
 }  
 if (isCorrect == 1) {  
 newWords[newSize] = new wchar\_t[wcslen(sentence.words[i]) + 1];  
 wcscpy(newWords[newSize], sentence.words[i]);  
 ++newSize;  
 }  
 }  
 }  
  
 for (int i = 0; i < sentence.size; ++i) {  
 delete[] sentence.words[i];  
 }  
  
 delete[] sentence.words;  
  
 sentence.words = newWords;  
 sentence.size = newSize;  
}

1. **Приложение (одно на бригаду)**

Наша программа состоит из основного и 3 заголовочных файлов:

* dz.cpp – основной файл, содержащий в себе функционал для взаимодействия пользователя с нашей программой (меню).
* Library.h – заголовочный файл, содержащий в себе библиотеку необходимых для работы программы функций.
* Sentence.h – заголовочный файл, содержащий в себе функции для заполнения введенного текста в структуру Sentence.
* Text.h – заголовочный файл, содержащий в себе функции для заполнения введенного текста в структуру Text.

dz.cpp:

#pragma once  
#pragma warning(disable : 4996)  
  
#include <iostream>  
#include <locale>  
#include <io.h>  
#include <fcntl.h>  
#include "Library.h"  
#include "Sentence.h"  
#include "Text.h"  
  
using namespace std;  
  
const int textLen = 5000;  
  
wchar\_t\* loadText() {  
 wchar\_t\* str = new wchar\_t[wcslen(testText) + 1];  
 for (int i = 0; i < wcslen(testText); ++i) {  
 str[i] = testText[i];  
 }  
 str[wcslen(str) + 1] = '\0';  
 return str;  
}  
  
wchar\_t\* textInput() {  
 wchar\_t\* str = new wchar\_t[textLen];  
 wcout << L"Введите текст:\n";  
 wchar\_t c;  
 int index = 0;  
 while ((c = \_getwch()) != wchar\_t(26)) {  
 if (c == wchar\_t(8)) {  
 wcout << L"\b \b";  
 --index;  
 }  
 else if (isCorrect(c)) {  
 str[index++] = c;  
 \_putwch(c);  
 }  
 else {  
 wcout << wchar\_t(7);  
 }  
 }  
  
 str[index] = '\0';  
  
 int size = wcslen(str) + 1;  
 wchar\_t\* new\_str = new wchar\_t[size];  
 for (int i = 0; i < index; ++i)  
 new\_str[i] = str[i];  
 new\_str[size - 1] = '\0';  
  
 delete[]str;  
 return new\_str;  
}  
  
void print(Text& text) {  
 for (int sentCount = 0; sentCount < text.size; sentCount++) {  
 for (int wordCount = 0; wordCount < text.sentences[sentCount].size; wordCount++) {  
 wcout << L"(" << text.sentences[sentCount].words[wordCount] << L")";  
 }  
 wcout << endl;  
 }  
 system("pause");  
}  
  
void variant14(Sentence& sentence) {  
// Оставить в предложениях слова,  
// отличающиеся от последнего слова предложения,  
// в которые каждая буква входит не менее двух раз.  
  
 int lastIndex = sentence.size - 1;  
 wchar\_t\*\* newWords = new wchar\_t\* [sentence.size];  
 int newSize = 0;  
  
 for (int i = 0; i <= lastIndex; ++i) {  
 if (wcscmp(sentence.words[lastIndex], sentence.words[i]) || i == lastIndex) {  
 bool isCorrect = true;  
 for (int j = 0; j < wcslen(sentence.words[i]); ++j) {  
 int count = 1;  
 for (int k = 0; k < wcslen(sentence.words[i]); ++k) {  
 if (k != j && sentence.words[i][k] == sentence.words[i][j])  
 ++count;  
 }  
 if (count < 2) {  
 isCorrect = false;  
 break;  
 }  
 }  
 if (isCorrect) {  
 newWords[newSize] = new wchar\_t[wcslen(sentence.words[i]) + 1];  
 wcscpy(newWords[newSize], sentence.words[i]);  
 ++newSize;  
 }  
 }  
 }  
  
 for (int i = 0; i < sentence.size; ++i) {  
 delete[] sentence.words[i];  
 }  
 delete[] sentence.words;  
  
 sentence.words = newWords;  
 sentence.size = newSize;  
}  
  
void variant15(Sentence& sentence) {  
// Оставить в предложениях слова,  
// отличающиеся от последнего слова предложения,  
// в которых гласные буквы (a, e, i, o, u, y) чередуются с согласными.  
  
 int LastWordIndex = sentence.size - 1;  
 wchar\_t\*\* newWords = new wchar\_t\*[sentence.size];  
 wchar\_t\* currentWord;  
 int newSize = 0;  
 bool isCorrect;  
 bool prevSoglas = true;  
 bool currSoglas = true;  
 char arr[] = {'a','e','i','o','u','y'};  
  
 for (int i = 0; i <= LastWordIndex; i++) {  
 if (wcscmp(sentence.words[LastWordIndex], sentence.words[i]) || i == LastWordIndex) {  
 isCorrect = true;  
 currentWord = sentence.words[i];  
 for (int f = 0; f < sizeof(arr)/sizeof(arr[0]); f++) {  
 if (currentWord[0] == arr[f]) {  
 prevSoglas = false;  
 }  
 }  
 for (int j = 1; j < wcslen(currentWord) - 1; j++) {  
 for (int k = 0; k < sizeof(arr)/sizeof(arr[0]); k++) {  
 if (currentWord[j] == arr[k]) {  
 currSoglas = false;  
 }  
 }  
 if (currSoglas == prevSoglas) {  
 isCorrect = false;  
 break;  
 }  
 else {  
 prevSoglas = currSoglas;  
 currSoglas = true;  
 }  
 }  
 if (isCorrect == 1) {  
 newWords[newSize] = new wchar\_t[wcslen(sentence.words[i]) + 1];  
 wcscpy(newWords[newSize], sentence.words[i]);  
 ++newSize;  
 }  
 }  
 }  
  
 for (int i = 0; i < sentence.size; ++i) {  
 delete[] sentence.words[i];  
 }  
  
 delete[] sentence.words;  
  
 sentence.words = newWords;  
 sentence.size = newSize;  
}  
  
void variant22(Sentence& sentence) {  
// Оставить в предложениях слова, отличающихся от последнего слова предложения,  
// заменив в них каждую букву, стоящую в алфавите на месте k,  
// на букву, стоящую в алфавите на месте k с конца алфавита (при замене сохранить регистр).  
  
 int LastWordIndex = sentence.size - 1;  
 wchar\_t\*\* newWords = new wchar\_t\*[sentence.size];  
 wchar\_t\* currentWord;  
 int newSize = 0, k;  
 bool isCorrect;  
  
 for (int i = 0; i <= LastWordIndex; ++i) {  
 if (wcscmp(sentence.words[LastWordIndex], sentence.words[i]) || i == LastWordIndex) {  
 isCorrect = true;  
 currentWord = sentence.words[i];  
 for (int j = 0; j < wcslen(currentWord); j++) {  
 if (currentWord[j] != '.' && currentWord[j] != '!' && currentWord[j] != '?') {  
 k = currentWord[j];  
 currentWord[j] = 122 - k + 97;  
 }  
 }  
 }  
 else  
 isCorrect = false;  
  
 if (isCorrect) {  
 newWords[newSize] = new wchar\_t[wcslen(sentence.words[i]) + 1];  
 wcscpy(newWords[newSize], sentence.words[i]);  
 ++newSize;  
 }  
 }  
  
 for (int i = 0; i < sentence.size; ++i)  
 delete[] sentence.words[i];  
 delete[] sentence.words;  
  
 sentence.words = newWords;  
 sentence.size = newSize;  
}  
  
short int menu() {  
 wcout << L"================== Домашняя работа ==================" << endl << endl;  
 wcout << L"\t1. Ввести текст" << endl;  
 wcout << L"\t2. Ввести стандартный пример текста" << endl;  
 wcout << L"\t3. Заполнить структуру Text" << endl;  
 wcout << L"\t4. Вывести текст" << endl << endl;  
 wcout << L"\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  
 wcout << L"\t|5. Индивидуальное задание (14 вариант)|" << endl;  
 wcout << L"\t|6. Индивидуальное задание (15 вариант)|" << endl;  
 wcout << L"\t|7. Индивидуальное задание (22 вариант)|" << endl;  
 wcout << L"\t‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾" << endl << endl;  
 wcout << L"\t8. Выход из программы" << endl << endl;  
 wcout << L"============= Бенц, Коновалов, Самойлов =============" << endl << endl;  
 wcout << L"Введите номер пункта меню: ";  
  
 short int choice;  
 wcin >> choice;  
  
 while (wcin.fail()) {  
 wcout << L"Введено неверное значение. Повторите попытку.\n";  
 wcin.clear();  
 wcin.ignore(10, '\n');  
 wcin >> choice;  
 }  
 return choice;  
}  
  
void outputSettings() {  
 \_setmode(\_fileno(stdout), \_O\_U16TEXT);  
 system("chcp 65001");  
 system("cls");  
}  
  
int main() {  
 wchar\_t\* input = nullptr;  
 Text text;  
 outputSettings();  
 while (true) {  
 short int choice = menu();  
 system("cls");  
 switch (choice) {  
 case 1:  
 input = textInput();  
 break;  
 case 2:  
 input = loadText();  
 break;  
 case 3:  
 text = stringToText(input);  
 break;  
 case 4:  
 print(text);  
 break;  
 case 5:  
 for (int i = 0; i < text.size; ++i)  
 variant14(text.sentences[i]);  
 break;  
 case 6:  
 for (int i = 0; i < text.size; ++i)  
 variant15(text.sentences[i]);  
 break;  
 case 7:  
 for (int i = 0; i < text.size; ++i)  
 variant22(text.sentences[i]);  
 break;  
 case 8:  
 return 0;  
 default:  
 break;  
 }  
 system("cls");  
 }  
}

Library.h:

#pragma once

#pragma warning(disable : 4996)

const wchar\_t punctuation[] = L" .!?,:;";

const wchar\_t\* testText = L"Hello, World! BMSTU is the best university in the world. Hi, what is your name?";

bool isPunctuation(wchar\_t c) {

for (const wchar\_t& i : punctuation) {

if (c == i)

return true;

}

return false;

}

bool isSentenceEnding(wchar\_t c) {

return (c == L'.') || (c == L'!') || (c == L'?');

}

bool isCorrect(wchar\_t c) {

return iswalpha(c) || isPunctuation(c);

}

wchar\_t\* substr(wchar\_t\* str, int start, int end) {

wchar\_t\* sub = new wchar\_t[end - start + 2];

for (int i = start; i <= end; ++i) {

sub[i - start] = str[i];

}

sub[end - start + 1] = L'\0';

return sub;

}

Sentence.h:

#pragma once

#pragma warning(disable : 4996)

#include "Library.h"

#include <iostream>

#include <conio.h>

struct Sentence {

wchar\_t\*\* words;//массив из указателей на слова в предложении

int size; //размер массива (число слов в предложении)

};

Sentence stringToSentence(const wchar\_t\* str) {

wchar\_t\* copy = \_wcsdup(str);

wchar\_t\*\* words = new wchar\_t \* [100];

int wordsAmount = 0;

wchar\_t\* token = NULL, \*nextToken = NULL;

token = wcstok\_s(copy, punctuation, &nextToken);

while (token != NULL) {

words[wordsAmount] = new wchar\_t[wcslen(token) + 1];

wcscpy\_s(words[wordsAmount], wcslen(token) + 1, token);

++wordsAmount;

token = wcstok\_s(NULL, punctuation, &nextToken);

}

free(copy);

return {words, wordsAmount};

}

Text.h:

#pragma once

#pragma warning(disable : 4996)

#include "Sentence.h"

#include "Library.h"

#include <iostream>

#include <wchar.h>

struct Text {

Sentence\* sentences;

int size;

};

Text stringToText(wchar\_t\* str) {

int sentencesAmount = 0;

int startingIndex = 0;

Sentence sentences[100];

for (int i = 0; i < wcslen(str); ++i) {

std::wcout << i << std::endl;

if (isSentenceEnding(str[i])) {

sentences[sentencesAmount] = stringToSentence(substr(str, startingIndex, i));

++sentencesAmount;

startingIndex = i + 1;

}

}

Text text;

text.size = sentencesAmount;

Sentence\* optimalSentences = new Sentence[sentencesAmount];

for (int i = 0; i < sentencesAmount; ++i) {

optimalSentences[i] = sentences[i];

}

text.sentences = optimalSentences;

return text;

}

**5. Список использованной литературы.**

Основная литература:

1. Р. Лафоре. Объектно-ориентированное программирование в С++. Издательство ПИТЕР, 2004 г.
2. Эккель Б. Философия C++. Введение в стандартный C++. 2-е изд. — СПб.: Питер, 2004. — 572 с.: ил.
3. Папшев И.С. Методические указания к ЛВП1, Электронная версия, 2020.

Дополнительная литература:

1. Н. Вирт. Алгоритмы + Структуры данных = Программы, «Мир», 1985 г.
2. Д. Ван Тассел. Стиль, разработка, эффективность, отладка и испытание программ: Пер. с англ. — 2-е изд., испр., — М.: Мир, 1985.